Slide 1.

WHAT IS REDIS?  
Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker.

To achieve top performance, Redis works with an in-memory dataset. Depending on your use case, you can persist your data either by periodically dumping the dataset to disk or by appending each command to a disk-based log. You can also disable persistence if you just need a feature-rich, networked, in-memory cache.

Slide 2

WHERE REDIS USED?

Due to the nature of the database design, typical use cases are session caching, full page cache, message queue applications, leaderboards and counting among others.

Slide 3

IS REDIS POPULAR?

Large companies such as Twitter are using Redis, Amazon Web Services offers a managed Redis service called Elasticache for Redis, Microsoft offers Azure Cache for Redis in Azure, and Alibaba is offering ApsaraDB for Redis in Alibaba Cloud.

Slide 4

EXAMPLE OF USING REDIS.

Elasticache for Redis by Amazon

[Image]

The cache-aside strategy is one of the most popular options for boosting database performance. When an application needs to read data from a database, it first queries the cache. If the data is not found, the application queries the database and populates the cache with the result. Cached data can get stale and we can set time to live for an it.

Slide 5

WHY REDIS?

* It`s open source and free.
* It`s very fast.

(Supports asynchronous replication).

* Provides data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes, and streams.

You can run atomic operations on these types, like appending to a string; incrementing the value in a hash; pushing an element to a list; computing set intersection, union and difference; or getting the member with highest ranking in a sorted set.

Slide 6

*Transactions.*  
They allow the execution of a group of commands in a single step, with two important guarantees:

All the commands in a transaction are serialized and executed sequentially. It can never happen that a request issued by another client is served in the middle of the execution of a Redis transaction. This guarantees that the commands are executed as a single isolated operation.

Either all of the commands or none are processed, so a Redis transaction is also atomic.

*Pub/Sub*  
SUBSCRIBE, UNSUBSCRIBE and PUBLISH implement the Publish/Subscribe messaging paradigm. Published messages are characterized into channels, without knowledge of what (if any) subscribers there may be. Subscribers express interest in one or more channels, and only receive messages that are of interest, without knowledge of what (if any) publishers there are. This decoupling of publishers and subscribers can allow for greater scalability and a more dynamic network topology.

For instance in order to subscribe to channels foo and bar the client issues a SUBSCRIBE providing the names of the channels:

SUBSCRIBE foo bar

Messages sent by other clients to these channels will be pushed by Redis to all the subscribed clients.

A client subscribed to one or more channels should not issue commands, although it can subscribe and unsubscribe to and from other channels

For example it can be used for web-chat.

*Automatic failover.*

If a master is not working as expected, Sentinel can start a failover process where a replica is promoted to master, the other additional replicas are reconfigured to use the new master, and the applications using the Redis server are informed about the new address to use when connecting.

Slide 7

WHICH OS SUPPORTS REDIS?

Redis is written in ANSI C and works in most POSIX systems like Linux, \*BSD, and OS X, without external dependencies. There is no official support for Windows builds, but we can use Redis in Open Server.

Slide 8

HOW TO INSTALL REDIS?

Download it from redis.io or use this special URL that always points to the latest stable Redis version, that is, <http://download.redis.io/redis-stable.tar.gz>

Slide 9

TO COMPILE REDIS FOLLOW THESE STEPS:

[code]

Slide 10

It is a good idea to copy both the Redis server and the command line interface into the proper places, either manually using the following commands.

[code]

Slide 11

NOW YOU CAN RUN REDIS

The simplest way to start the Redis server is just executing the redis-server binary without any argument.

Slide 12

Redis provides a command line utility that can be used to send commands to Redis. This program is called redis-cli. The first thing to do in order to check if Redis is working properly is sending a PING command using redis-cli.

Command name is PING.

When Redis answer is PONG it mean that all goes right.

Slide 12

LIST OF FREQUENTLY USED COMMANDS:

GET - Get key value

SET - Set the string value of a key

EXISTS - Determine if key exists

FLUSHALL - Delete all the keys of all the existing databases, not just the currently selected one

GETSET - Set the string value of a key and return its old value

DEL - Delete a key

KEYS - Find all keys matching the giving pattern

INCR / DECR - If value of key is integer - icrement / decriment it by one

TTL - Get the time to live for a key

PERSIST - Remove the expiration from a key

RENAME - Rename a key

Slide 13

FULL LIST OF REDIS COMMANDS: REDIS.IO/COMMANDS

Slide 14

YOU CAN TRY TO WORK WITH REDIS ON THIS PAGE:

TRY.REDIS.IO

Slide 15

LINKS

redis.io

redis.io/documentation

aws.amazon.com/ru/redis/

Slide 16

THANKS FOR YOUR ATTENTION!